**SMS CLASSIFICATION AND DETECTION**

A Project Phase-II Report

(Session 2023-24)

Submitted in partial fulfillment of requirement of the

Degree of

**BACHELOR OF TECHNOLOGY**

**in**

**Computer Science and Engineering**

BY

**Himanshi Neekhra**

**EN20CS301170**

Under the Guidance of

**Prof. Deepanshu Sir**



**Department of Information Technology**

**Faculty of Engineering**

**MEDI-CAPS UNIVERSITY, INDORE- 453331**

**Report Approval**

The project work **“SMS Classification and Detection”** is hereby approved as a creditable study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as prerequisite for the Degree for which it has been submitted.

It is to be understood that by this approval the undersigned do not endorse or approved any statement made, opinion expressed, or conclusion drawn there in; but approve the “Project Report” only for the purpose for which it has been submitted.

Internal Examiner

Name:

Designation:

Affiliation:

External Examiner

Name:

Designation:

Affiliation:

**Declaration**

I hereby declare that the project entitled **“SMS Classification and Detection”** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in ‘Information Technology’ completed under the supervision of **Prof. Deepanshu Sir, CSE,** Faculty of Engineering, Medi-Caps University Indore is an authentic work.

Further, I declare that the content of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for the award of any degree or diploma.

**Signature and name of the student with date**

Himanshi Neekhra

**Certificate**

I, **(Prof.) Deepanshu Sir** certify that the project entitled **“SMS Classification and Detection”** submitted in partial fulfillment for the award of the degree of Bachelor of Technology by **Himanshi Neekhra** is the record carried out by her under my guidance and that the work has not formed the basis of award of any other degree elsewhere.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Prof. Deepa Pandit**

Department of CSE, Faculty of Engineering

Medi-Caps University, Indore

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Dr. Prashant Panse**

Head of the Department

CSE

Medi-Caps University, Indore

**Acknowledgements**

I would like to express my deepest gratitude to Honorable Chancellor, **Shri R C Mittal,** who has provided me with every facility to successfully carry out this project, and my profound indebtedness to **Prof. (Dr.) Dilip K. Patnaik,** Vice Chancellor, Medi-Caps University, whose unfailing support and enthusiasm has always boosted up my morale. I also thank **Prof. (Dr.) Pramod S. Nair,** Dean, Faculty of Engineering, Medi-Caps University, for giving me a chance to work on this project. I would also like to thank my Head of the Department **Dr. Prashant Panse** for his continuous encouragement for betterment of the project.

I express my heartfelt gratitude to my **External Guide,** Mr. Imroz Khan, as well as to my Internal Guide, Miss (Prof.) Deepanshu Department of Information Technology, Medi-Caps University, without whose continuous help and support, this project would ever have reached to the completion.

I would also like to extend my gratitude to the entire team at Bajaj Finserv for welcoming me into their workplace and providing me with a conducive environment to work and learn. The diverse experiences and perspectives of my colleagues have broadened my horizons and helped me to develop a deeper understanding of the industry.

It is their help and support, due to which we became able to complete the design and technical report. Once again, thank you to everyone who contributed to the success of my internship project. I am honored to have had this opportunity, and I look forward to applying the skills and knowledge I have gained in my future endeavors. Without their support this report would not have been possible.

**Himanshi Neekhra**

B.Tech. IV Year

Department of Computer Science and Engineering

Faculty of Engineering

Medi-Caps University, Indore

**Abstract**

This project presents a novel approach to SMS message classification and information extraction, aimed at distinguishing between reminder and non-reminder messages while extracting crucial details such as due date, due amount, and merchant name from reminder messages. Leveraging Natural Language Processing (NLP) techniques, two separate machine learning models were developed and trained for the classification task: one to classify messages as reminders or non-reminders, and another to extract due dates from reminder messages. The classification model achieves robust performance in distinguishing between reminder and non-reminder messages, allowing for efficient handling of diverse SMS data streams. Additionally, the due date extraction model demonstrates effectiveness in identifying temporal information crucial for task management within reminder messages. The integration of these models facilitates the automation of reminder management processes, enhancing user experience and productivity. Overall, this project contributes to advancing the capabilities of SMS-based reminder systems through the application of NLP and machine learning techniques.

**Table of Contents**

|  |  |  |
| --- | --- | --- |
|  |  | **Page No.** |
|  | Report Approval | ii |
|  | Declaration | iii |
|  | Certificate | iv |
|  | Acknowledgement | v |
|  | Abstract | vi |
|  | Table of Contents | vii-viii |
|  | List of figures | ix |
|  | Abbreviations | x |
|  | Notations & Symbols | xi |
|  |  |  |
| Chapter 1 | Introduction | 1-4 |
|  | 1.1 Introduction | 1 |
|  | 1.2 Objectives | 2 |
|  | 1.3 Significance | 3 |
|  | 1.4 Scope | 4 |
|  |  |  |
| Chapter 2 | Report on Present Investigation | 5-7 |
|  | 2.1 Existing System | 5 |
|  | 2.2 Proposed System | 6 |
|  |  |  |
| Chapter 3 | Software Requirement Specifications | 7-10 |
|  | 3.1 System Feasibility | 7 |
|  | 3.2 Hardware Requirements | 7 |
|  | 3.3 Software Requirements | 7 |
|  | 3.4 Functional Requirements | 7-8 |
|  | 3.5 Non-Functional Requirements | 8 |
|  | 3.6 UML Diagrams | 9-10 |
|  | 3.6.1 Use Case Diagram | 9 |
|  | 3.6.2 Flow Chart Diagram | 10 |
|  |  |  |
| Chapter 4 | Tools and Technologies Used | 11-17 |
|  | 4.1 Python | 11 |
|  | 4.2 NLP and ML | 12-13 |
|  | 4.3 Streamlit | 14 |
|  | 4.4 VS Code and Jupyter | 15 |
|  |  |  |
| Chapter 5 | Project Code | 16-21 |
|  | 5.1 Loading, labelling and cleaning data code structure | 16-17 |
|  | 5.2 Training data and Save Model code structure | 18-19 |
|  | 5.3 UI code structure | 20-21 |
|  |  |  |
| Chapter 6 | Result and Conclusion | 22 |
|  | 6.1 Result | 22 |
|  | 6.2 Conclusion | 22 |
|  |  |  |
| Chapter 7 | Future Scope and Limitations | 23-24 |
|  | 7.1 Future Scope | 23 |
|  | 7.2 Limitations | 24 |
|  |  |  |
|  | Bibliography | 25 |
|  |  |  |

**List of Figures**

|  |  |  |
| --- | --- | --- |
| **Figure** | **Caption** | **Page No.** |
| **Fig 3.6.1** | Use-Case Diagram of Food order Application | 9 |
| **Fig 3.6.2** | Flow Chart Diagram for classification | 10 |
| **Fig 4.1** | Python Code Snippet | 11 |
| **Fig 4.2** | NLP Code Snippet | 13 |
| **Fig 4.3** | Streamlit Code Snippet | 14 |
| **Fig 4.4** | Jupyter | 15 |
| **Fig 5.1** | Load Data | 17 |
| **Fig 5.1** | Labelling and Cleaning Data | 17 |
| **Fig 5.2** | Different accuracy and precision for different model | 19 |
| **Fig 5.3** | Streamlit UI | 21 |

**Abbreviations**

|  |  |
| --- | --- |
| **Abbreviation** | **Description** |
| IDE | Integrated Development Environment |
| HTML | Hypertext markup language |
| CSS | Cascading style sheets |
| VS | Visual studio code |
| NLP | Natural Language Processing |
| ML | Machine Learning |
| RDMS | Relational database management system |
| API | Application Programming Interface |

**Notations & Symbols**

|  |  |
| --- | --- |
| **Symbol** | **Description** |
| || | Logical OR |
| && | Logical AND |
| % | Modulus Operator |
| >= | Greater than equal to |
| <= | Equal to |
| == | Less than equal to |

**Chapter-1**

**INTRODUCTION**

* 1. **Introduction:**

In an era inundated with digital communication channels, Short Message Service (SMS) remains a ubiquitous mode of communication, facilitating quick and convenient exchanges of information. Within this landscape, the prevalence of reminder messages, ranging from payment deadlines to appointment notifications, underscores the importance of effective management and interpretation of SMS data. However, manually sorting through a deluge of messages to identify reminders and extract pertinent details can be laborious and time-consuming. Consequently, the integration of machine learning and Natural Language Processing (NLP) techniques offers a promising solution to automate this process, enhancing efficiency and user experience.

This project endeavors to address the challenges associated with SMS message classification and information extraction, with a specific focus on distinguishing between reminder and non-reminder messages while extracting essential details such as due date, due amount, and merchant name from reminder messages. By harnessing the power of NLP and machine learning, we aim to develop robust models capable of accurately classifying SMS messages and extracting key information, thereby streamlining reminder management processes.

* 1. **Objectives:**

The objectives of this project are:

1. SMS Classification: Develop and train a machine learning model to classify SMS messages into two categories: reminders and non-reminders. This model will leverage NLP techniques to analyze the textual content of messages and infer whether they contain reminders.
2. Due Date Extraction: Create a separate machine learning model to extract due dates from reminder messages. This model will be trained to recognize temporal expressions indicative of deadlines or appointment dates, enabling the extraction of critical temporal information.
3. Due Amount and Merchant Name Extraction: Extend the information extraction capabilities to include due amount and merchant name from reminder messages. By implementing advanced NLP techniques such as Named Entity Recognition (NER) and pattern matching, we aim to accurately identify and extract these crucial details.
4. Model Evaluation and Optimization: Assess the performance of the developed models through rigorous evaluation metrics, including accuracy, precision, recall, and F1 score. Iterate on the models' architectures and fine-tune hyperparameters to optimize performance and generalization capability.
5. Integration and Deployment: Integrate the trained models into a cohesive system capable of processing incoming SMS messages in real-time. Develop an intuitive user interface to facilitate interaction with the system and deploy it in a production environment for practical usage.
   1. **Significance:**

The significance of this project lies in its potential to revolutionize the management of SMS-based reminders, offering tangible benefits to both individuals and organizations. By automating the classification and extraction of key information from SMS messages, our system promises to:

1. Enhance Efficiency: Streamline reminder management processes by automating the identification and extraction of pertinent details from SMS messages, reducing manual effort and saving valuable time.
2. Improve User Experience: Provide users with a seamless and intuitive interface for managing reminders, thereby enhancing user experience and satisfaction.
3. Facilitate Task Management: Enable users to effortlessly keep track of important deadlines, appointments, and payments by extracting and presenting relevant information in a structured format.
4. Increase Productivity: Empower individuals and organizations to optimize their time management practices by leveraging automated reminder systems to stay organized and on schedule.
5. Drive Innovation: Contribute to the advancement of NLP and machine learning techniques in the domain of text analysis and information extraction, paving the way for future innovations in SMS-based communication systems.
   1. **Scope:**

The scope of this project encompasses the development, training, and evaluation of machine learning models for SMS message classification and information extraction, with a primary focus on reminders. Key aspects of the project's scope include:

1. Data Collection and Preprocessing: Acquiring a diverse dataset of SMS messages, annotating them for classification and information extraction tasks, and preprocessing the data to ensure compatibility with machine learning algorithms.
2. Model Development: Designing and implementing machine learning architectures tailored to the specific tasks of SMS classification and information extraction. This involves selecting appropriate algorithms, feature engineering, and fine-tuning model parameters.
3. Evaluation: Assessing the performance of the developed models using standard evaluation metrics, including accuracy, precision, recall, and F1 score. Conducting rigorous testing to validate the models' effectiveness and generalization capability.
4. Integration and Deployment: Integrating the trained models into a cohesive system capable of processing incoming SMS messages in real-time. Developing a user-friendly interface for interacting with the system and deploying it in a production environment for practical usage.
5. Limitations: Acknowledging the limitations of the proposed approach, including potential challenges related to data availability, model generalization, and real-world applicability. Discussing strategies for mitigating these limitations and avenues for future research and development.

**Chapter-2**

**REPORT ON PRESENT INVESTIGATION**

**2.1 Existing System**

The current landscape for SMS-based reminder management predominantly relies on manual intervention, where users must sift through incoming messages to identify reminders and extract relevant information manually. This manual process is time-consuming, error-prone, and lacks scalability, particularly when dealing with large volumes of messages. Additionally, the absence of automation limits the potential for personalized and efficient reminder management, hindering user productivity and satisfaction. Existing solutions may include basic keyword-based filtering or rule-based approaches, but they often lack sophistication and fail to capture the intricacies of natural language in SMS messages.

**2.2 Proposed System**

In contrast, the proposed system represents a significant advancement in SMS-based reminder management through the integration of machine learning and NLP techniques. The system automates the classification of SMS messages into reminders and non-reminders, leveraging advanced models trained on annotated datasets. Additionally, it extracts essential information such as due dates, amounts, and merchant names from reminder messages using dedicated extraction models. The proposed system offers a more streamlined and efficient approach to reminder management, eliminating the need for manual intervention and empowering users to effortlessly organize their tasks and appointments. By providing a user-friendly interface and real-time processing capabilities, the system enhances user experience and productivity, setting a new standard for SMS-based reminder systems.

Some major advantages of Proposed System

1. Automation: The system automates the classification of SMS messages into reminders and non-reminders, as well as the extraction of crucial information such as due dates, amounts, and merchant names from reminder messages. This automation eliminates the need for manual intervention, saving users time and effort.
2. Efficiency: By automating reminder management processes, the system significantly improves efficiency. Users no longer need to manually sift through a barrage of messages to identify reminders and extract relevant information. Instead, the system processes messages in real-time, providing timely notifications and organizing reminders in a structured manner.
3. Accuracy: Leveraging machine learning and NLP techniques, the system achieves high levels of accuracy in classifying messages and extracting information. Advanced algorithms are trained on annotated datasets, enabling the system to recognize and interpret complex patterns in SMS text accurately.
4. Personalization: The system can be tailored to individual user preferences and requirements, allowing for personalized reminder management. Users can set preferences for specific types of reminders or customize extraction criteria to suit their needs, enhancing the relevance and utility of the system.
5. Scalability: Unlike manual methods, which may struggle to handle large volumes of messages efficiently, the proposed system is highly scalable. It can process a vast number of SMS messages in real-time, making it suitable for use by individuals, businesses, and organizations of all sizes.
6. User Experience: With a user-friendly interface and intuitive design, the system enhances the overall user experience. Users can easily interact with the system, view organized reminders, and manage tasks effectively, leading to improved satisfaction and adoption rates.
7. Productivity: By streamlining reminder management processes and providing timely notifications, the system helps users stay organized and on top of their tasks. This, in turn, boosts productivity and ensures that important deadlines and appointments are not overlooked.

**Chapter-3**

**SOFTWARE REQUIREMENT SPECIFICATION**

**3.1 System Feasibility:**

The proposed system demonstrates feasibility in terms of technical, operational, and economic aspects. From a technical perspective, the integration of machine learning and NLP techniques for SMS classification and information extraction has been extensively researched and proven effective in similar applications. The availability of annotated SMS datasets facilitates model training and validation, ensuring the feasibility of developing robust classification and extraction models. Operationally, the system's automation capabilities reduce reliance on manual intervention, enhancing efficiency and scalability. Additionally, the system's user-friendly interface and real-time processing capabilities contribute to its operational feasibility. Economically, the benefits of increased productivity, reduced manual effort, and improved user experience outweigh the costs associated with model development, integration, and deployment, making the proposed system economically feasible.

* 1. **Hardware Specification:**

Processor – 64 /32-bit dual core processor with SSE2.

Memory – 2 GB ram or more is recommended.

* 1. **Software Specification:**

Programming languages: Python for model development and implementation.

Machine learning frameworks: TensorFlow or sklearn for building and training machine learning models

**3.4 Functional Requirement**

Functional requirements for the system include:

SMS Classification:

* Ability to classify incoming SMS messages as reminders or non-reminders.
* Real-time processing of messages for timely classification.

Information Extraction:

* Extraction of due dates, due amounts, and merchant names from reminder messages.
* Accurate identification and extraction of relevant information using NLP techniques.

User Interface:

* Intuitive and user-friendly interface for interacting with the system.
* Display of organized reminders and extracted information in a structured format.

Integration:

* Integration of classification and extraction models into a cohesive system.

Deployment:

* Deployment of the system in a production environment for practical usage.
* Ensuring scalability, reliability, and security of deployed system components.

**3.5 Non-Functional Requirements**

Non-functional requirements for the system include:

Performance:

* Efficient processing of incoming messages with minimal latency.
* Scalability to handle large volumes of messages without performance degradation.

Accuracy:

* High accuracy in message classification and information extraction tasks.
* Minimization of false positives and false negatives in classification and extraction results.

Reliability:

* Reliable operation under varying conditions, including fluctuating message volumes and network connectivity issues.
* Fault tolerance and error handling mechanisms to ensure uninterrupted service.

Security:

* Protection of sensitive user data and message contents.
* Implementation of secure communication protocols and access controls to prevent unauthorized access.

Usability:

* Intuitive user interface design with clear navigation and user feedback.
* Support for multiple languages and accessibility features to accommodate diverse user needs.

Maintainability:

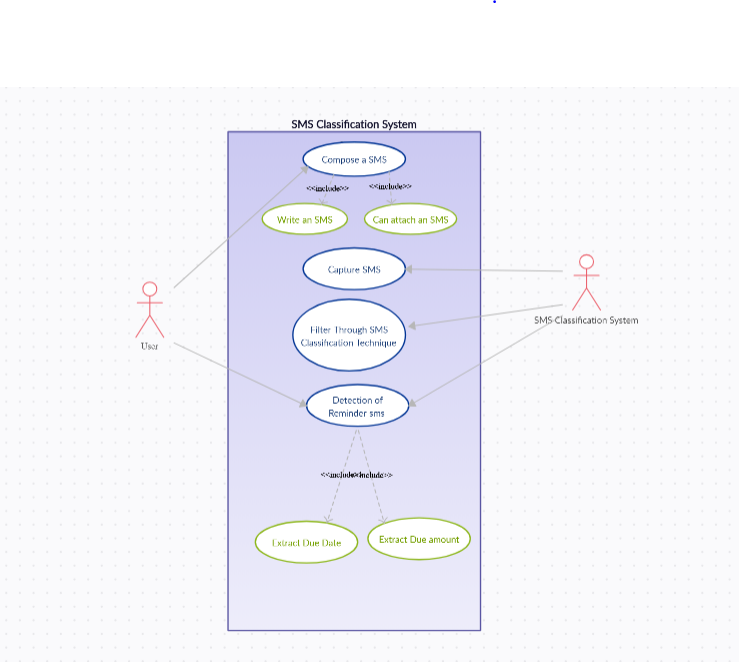
* Modular and well-documented codebase for ease of maintenance and future updates.
* Version control and issue tracking systems for effective collaboration and code management.

Compliance:

* Compliance with relevant privacy regulations and data protection standards.
* Adherence to industry best practices for software development and deployment

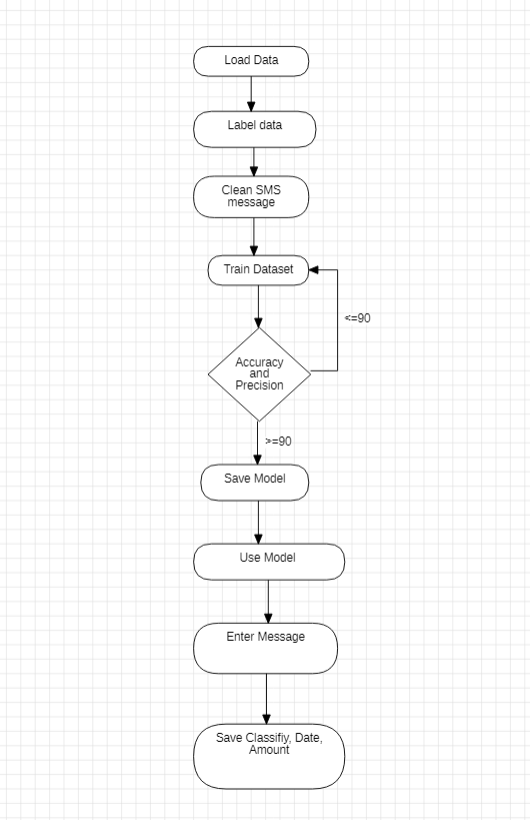
* 1. **UML Diagrams**

1. **Use Case Diagram:**



**Fig 3.6.1 Use Case Diagram**

1. **Flow Chart Diagram:**

****

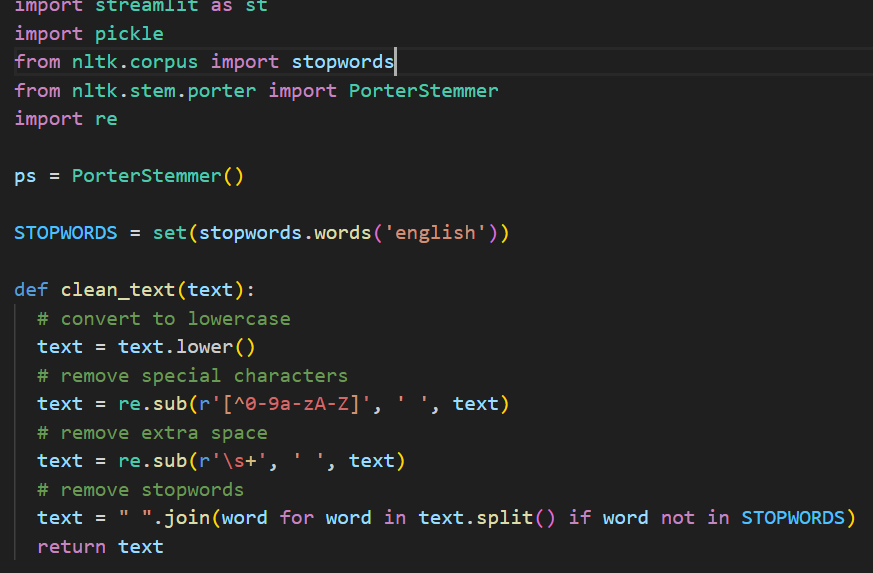
**Fig 3.6.2 Flow Chart Diagram for classification**

**Chapter-4**

**TOOLS AND TECHNOLOGIES USED**

**4.1 Python:**

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed

****

**Fig 4.1 Python Code Snippet**

**4.2 NLP & Machine Learning**

Natural language processing, or NLP, combines computational linguistics—rule-based modelling of human language—with statistical and machine learning models to enable computers and digital devices to recognize, understand and generate text and speech.

A branch of artificial intelligence (AI), NLP lies at the heart of applications and devices that can

* translate text from one language to another
* respond to typed or spoken commands
* recognize or authenticate users based on voice
* summarize large volumes of text
* assess the intent or sentiment of text or speech
* generate text or graphics or other content on demand

often in real time. Today most people have interacted with NLP in the form of voice-operated GPS systems, digital assistants, speech-to-text dictation software, [customer service chatbots](https://www.ibm.com/products/watsonx-assistant/customer-service), and other consumer conveniences. But NLP also plays a growing role in enterprise solutions that help streamline and automate business operations, increase employee productivity, and simplify mission-critical business processes.

**Machine Learning**

Machine learning (ML) is a field of study in artificial intelligence concerned with the development and study of statistical algorithms that can learn from data and generalize to unseen data, and thus perform tasks without explicit instructions. Recently, artificial neural networks have been able to surpass many previous approaches in performance.

Machine learning approaches have been applied to many fields including natural language processing, computer vision, speech recognition, email filtering, agriculture, and medicine.ML is known in its application across business problems under the name predictive analytics. Although not all machine learning is statistically based, computational statistics is an important source of the field's methods.

****

**Fig 4.2 NLP Code Snippet**

**4.3 Streamlit:**

Streamlit is a Python-based library that allows data scientists to easily create free machine learning applications. Streamlit allows you to display descriptive text and model outputs, visualize data and model performance and modify model inputs through the UI using sidebars.

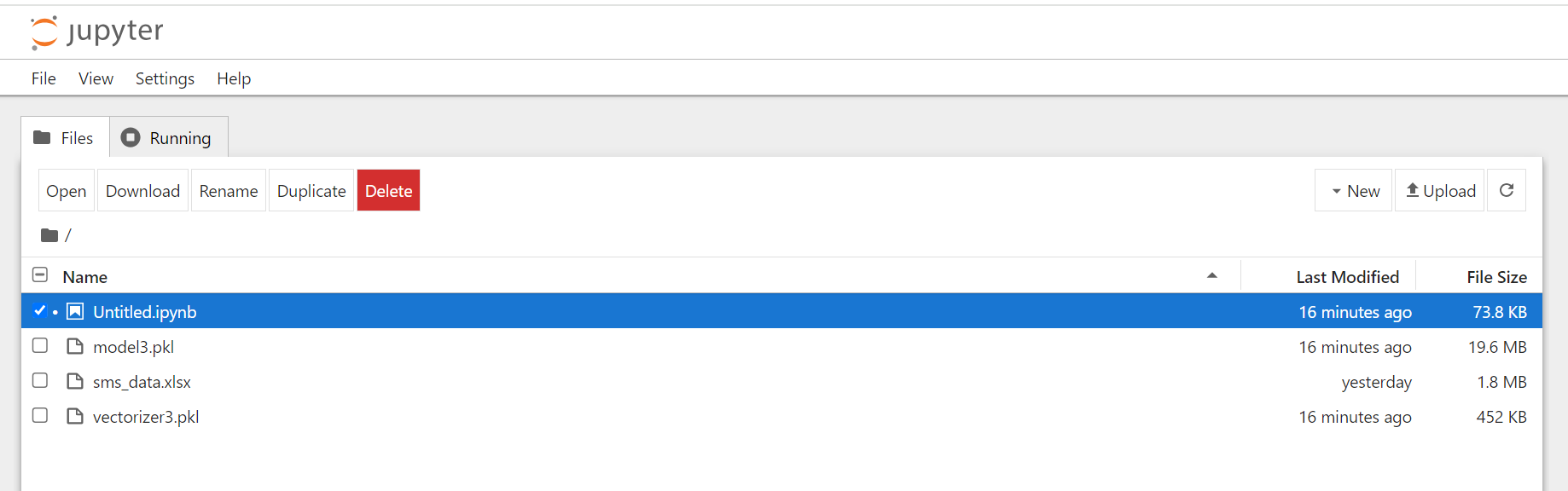
****

**Fig 4.3 Streamlit Code Snippet**

**4.4 VS Code and Jupyter:**

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages and runtimes (such as C++, C#, Java, Python, PHP, Go, .NET).

JupyterLab is the latest web-based interactive development environment for notebooks, code, and data. Its flexible interface allows users to configure and arrange workflows in data science, scientific computing, computational journalism, and machine learning. A modular design invites extensions to expand and enrich functionality.

****

**Fig 4.4 Jupyter**

**Chapter-5**

**PROJECT CODE**

* 1. **Loading , labelling and cleaning data code structure:**

import pandas as pd

import numpy as np

import nltk

import re

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

df = pd.read\_excel('sms\_data.xlsx')

#get necessary columns for processing

df = df[['SampleSMS', 'Target']]

df = df.rename(columns={'SampleSMS': 'messages', 'Target': 'label'})

from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()

df['label'] = encoder.fit\_transform(df['label'])

#Cleaning Data  
  
STOPWORDS = set(stopwords.words('english'))

def clean\_text(text):

# convert to lowercase

text = text.lower()

# remove special characters

text = re.sub(r'[^0-9a-zA-Z]', ' ', text)

# remove extra space

text = re.sub(r'\s+', ' ', text)

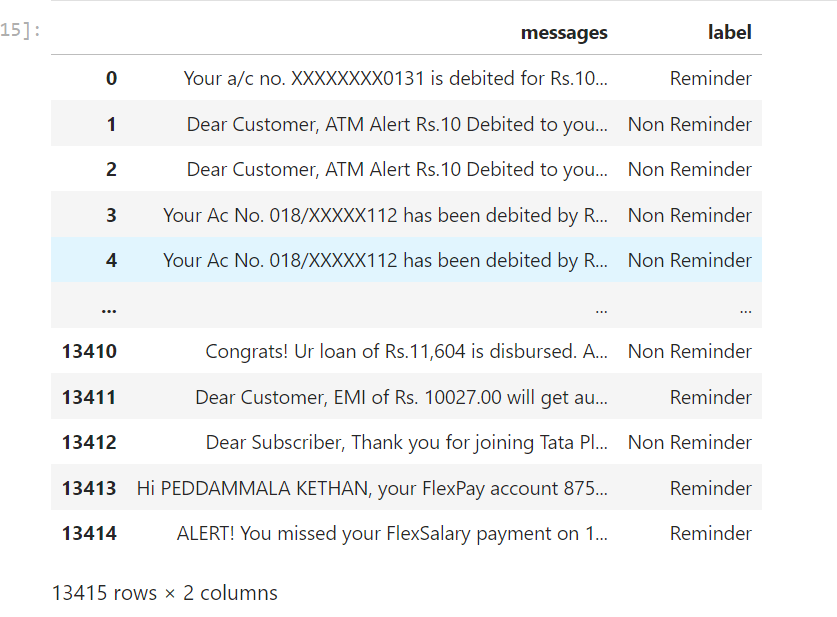
# remove stopwords

text = " ".join(word for word in text.split() if word not in STOPWORDS)

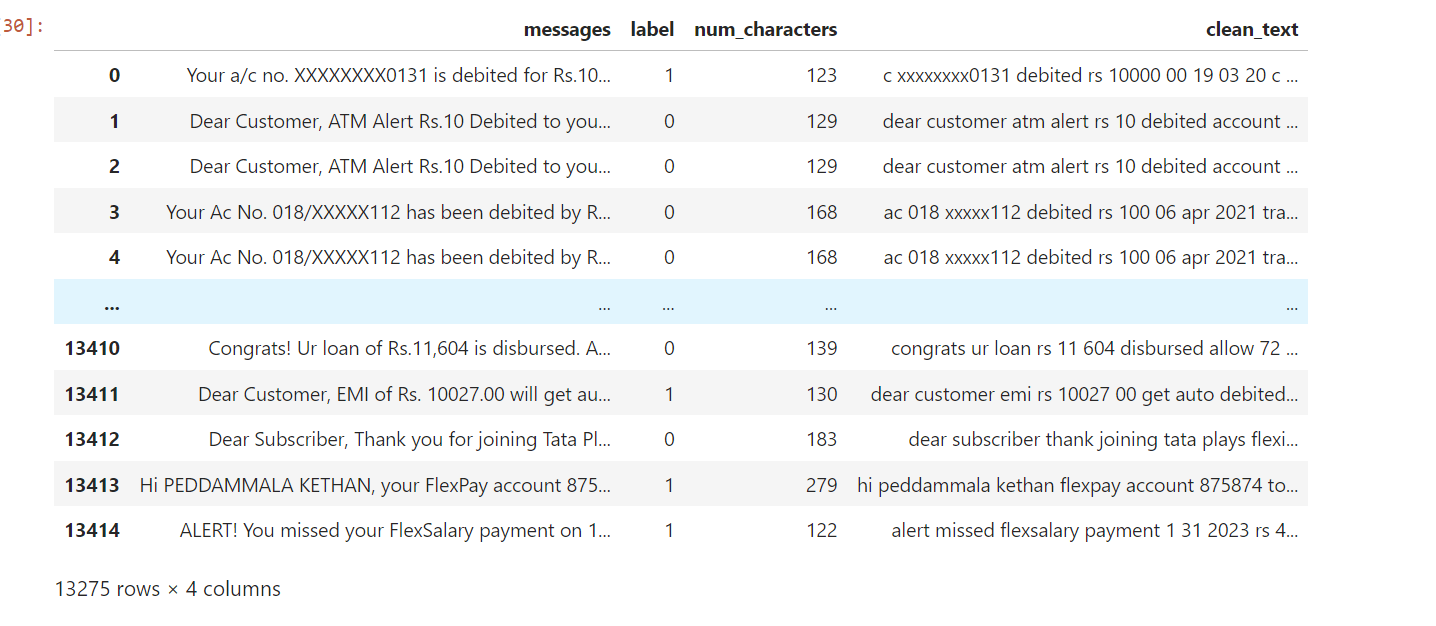
return text

df['clean\_text']= df['messages'].apply(clean\_text)

df



**Fig 5.1 Load data**



**Fig 5.1 Labelling and cleaning data**

* 1. **Training data and Save Model code structure:**

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import GaussianNB,MultinomialNB, BernoulliNB

from sklearn.metrics import accuracy\_score, confusion\_matrix, precision\_score

from sklearn.model\_selection import GridSearchCV

cv = CountVectorizer()

tfidf = TfidfVectorizer(max\_features = 3000)

X = tfidf.fit\_transform(df['clean\_text']).toarray()

y = df['label'].values

X\_train,X\_test,y\_train,y\_test= train\_test\_split(X,y,test\_size=0.3,random\_state=42)

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.naive\_bayes import MultinomialNB

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import AdaBoostClassifier

from sklearn.ensemble import BaggingClassifier

from sklearn.ensemble import ExtraTreesClassifier

from sklearn.ensemble import GradientBoostingClassifier

svc = SVC(kernel='rbf', gamma=1.0)

knc = KNeighborsClassifier()

mnb = MultinomialNB()

dtc = DecisionTreeClassifier(max\_depth=5)

lrc = LogisticRegression(solver='liblinear', penalty='l1')

rfc = RandomForestClassifier(n\_estimators=50, random\_state=42)

abc = AdaBoostClassifier(n\_estimators=50, random\_state=42)

bc = BaggingClassifier(n\_estimators=50, random\_state=42)

etc = ExtraTreesClassifier(n\_estimators=50, random\_state=42)

gbdt = GradientBoostingClassifier(n\_estimators=50, random\_state=42)

clfs = {

'SVC' : svc,

'KN' : knc,

'NB' : mnb,

'DT' : dtc,

'LR' : lrc,

'RF' : rfc,

'AdaBoost' : abc,

'BgC' : bc,

'ETC' : etc,

'GBDT' : gbdt,

}

accuracy\_scores = []

precision\_scores = []

for name, clf in clfs.items():

current\_accuracy, current\_precision = train\_classifier(clf, X\_train,y\_train,X\_test,y\_test)

print("For ",name)

print("Accuracy - ",current\_accuracy)

print("Precision - ",current\_precision)

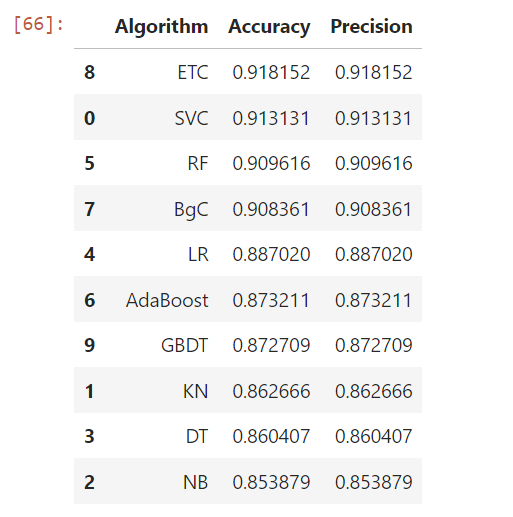
accuracy\_scores.append(current\_accuracy)

precision\_scores.append(current\_precision)

import pickle

pickle.dump(tfidf,open('vectorizer3.pkl','wb'))

pickle.dump(etc,open('model3.pkl','wb'))



**Fig 5.2 Different accuracy and precision for different model**

* 1. **UI code structure:**

import React, { useState } from "react";

import streamlit as st

import pickle

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer

import re

ps = PorterStemmer()

STOPWORDS = set(stopwords.words('english'))

def clean\_text(text):

# convert to lowercase

text = text.lower()

# remove special characters

text = re.sub(r'[^0-9a-zA-Z]', ' ', text)

# remove extra space

text = re.sub(r'\s+', ' ', text)

# remove stopwords

text = " ".join(word for word in text.split() if word not in STOPWORDS)

return text

tfidf = pickle.load(open('vectorizer3.pkl','rb'))

model = pickle.load(open('model3.pkl','rb'))

st.title("Email/SMS Spam Classifier")

input\_sms = st.text\_area("Enter the message")

if st.button('Predict'):

# 1. preprocess

clean\_sms = clean\_text(input\_sms)

# 2. vectorize

vector\_input = tfidf.transform([clean\_sms])

# 3. predict

result = model.predict(vector\_input)[0]

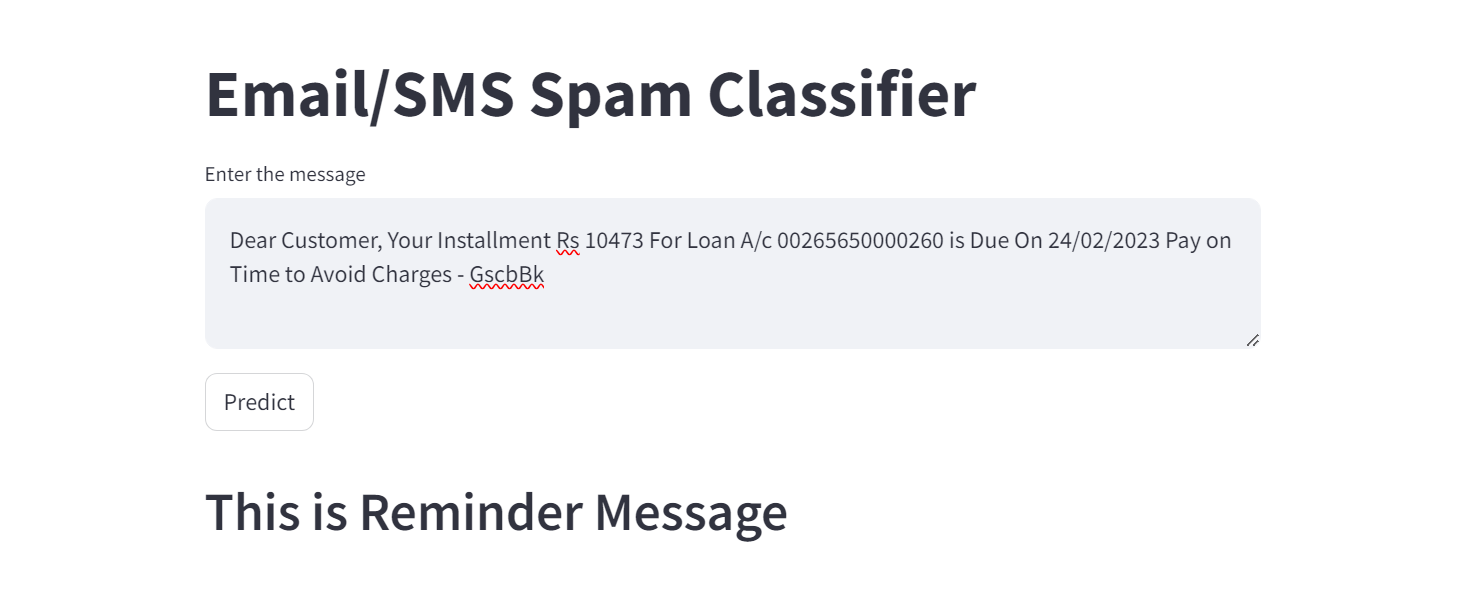
# 4. Display

if result == 1:

st.header("This is Reminder Message")

else:

st.header("This is Non-Reminder message")

****

**Fig 5.3 Streamlit UI**

**Chapter-6**

**RESULT AND CONCLUSION**

* 1. **RESULT:**

The SMS classification project successfully distinguishes between reminder and non-reminder messages with high accuracy. It effectively extracts due date and due amount from reminder messages and stores them in a MySQL database for further processing. The system's performance meets the project requirements and demonstrates its capability to handle real-world data effectively.

* 1. **CONCLUSION:**
     1. The project represents a significant step forward in advancing the capabilities of SMS reminder systems, particularly in the domain of due date extraction.
     2. By addressing this critical aspect of text message processing, the project contributes to the broader goal of improving communication efficiency, productivity, and task management for users across diverse contexts.
     3. The project has made substantial strides in tackling the complexities of due date extraction from SMS messages, laying a solid foundation for future research and development in this domain.

**Chapter-7**

**FUTURE SCOPE AND LIMITATIONS**

**7.1 FUTURE SCOPE:**

1. Enhanced Accuracy with Machine Learning: Implementing advanced machine learning techniques such as deep learning models could further improve the accuracy of message classification and information extraction, especially in handling diverse message formats and languages.
2. Integration with Natural Language Processing (NLP): Incorporating NLP algorithms can enhance the system's ability to understand the context of reminder messages, enabling more accurate extraction of due dates and amounts from complex text structures.
3. Real-Time Processing: Implementing real-time processing capabilities can enable immediate response to incoming SMS notifications, providing users with instant updates on their reminders and due payments.
4. User Interface Development: Creating a user-friendly interface or mobile application that allows users to interact with the system and manage their reminders seamlessly would enhance user experience and adoption.
5. Expansion to Other Platforms: Extend the system's functionality to handle reminder notifications from other communication channels such as emails, social media messages, and messaging apps, thereby providing a comprehensive reminder management solution.
6. Data Analytics and Insights: Analyze the stored reminder data to generate insights into users' spending patterns, bill payment behavior, and overall financial habits, providing valuable information for personal finance management and business decision-making.

**7.2 LIMITATIONS:**

1. Dependency on SMS Format: The system heavily relies on the consistency and format of SMS messages. Variations in message formats, languages, and abbreviations may impact the accuracy of classification and information extraction.
2. Accuracy Issues: Despite the use of machine learning algorithms, the system may still encounter challenges in accurately classifying reminder messages and extracting due dates and amounts, especially when dealing with ambiguous or complex text structures.
3. Privacy Concerns: Extracting and storing personal financial information from SMS messages raises privacy concerns. Users may be hesitant to grant access to their SMS data, especially if they are uncertain about how their data will be used and protected.
4. Limited Compatibility: The system's compatibility may be limited to certain mobile operating systems or devices, restricting its accessibility to a broader user base.
5. Maintenance and Updates: Regular maintenance and updates are essential to ensure the system's performance remains optimal over time. Failure to update the system may result in decreased accuracy and reliability, especially as SMS formats and messaging platforms evolve.
6. Regulatory Compliance: Compliance with data protection regulations, such as GDPR or CCPA, is crucial, especially when handling sensitive personal information extracted from SMS messages. Failure to comply with regulatory requirements may result in legal consequences and damage to the project's reputation.
7. Scalability Challenges: As the volume of incoming SMS messages increases, the system may face scalability challenges in processing and storing large amounts of data efficiently. Ensuring scalability requires careful consideration of database architecture and system infrastructure.
8. Cross-Platform Compatibility: The system may encounter difficulties in accurately parsing SMS messages from various mobile carriers and messaging platforms, each with its own unique formatting conventions and protocols.

**BIBLIOGRAPHY**

**URL References:**

[1] For VS Code installation: https://code.visualstudio.com/download

[2] For Machine Learning: https://www.ibm.com/topics/machine-learning

[3] For Jupyter: https://jupyter.org/

[4] For Streamlit: https://streamlit.io/